Auton Robot
DOI 10.1007/s10514-013-9363-y

Autonomously learning to visually detect where manipulation will succeed

Hai Nguyen · Charles C. Kemp

Received: 31 December 2012 / Accepted: 8 August 2013

Abstract Visual features can help predict if a manipulation behavior will succeed at a given location. For example, the success of a behavior that flips light switches depends on the location of the switch. We present methods that enable a mobile manipulator to autonomously learn a function that takes an RGB image and a registered 3D point cloud as input and returns a 3D location at which a manipulation behavior is likely to succeed. With our methods, robots autonomously train a pair of support vector machine (SVM) classifiers by trying behaviors at locations in the world and observing the results. Our methods require a pair of manipulation behaviors that can change the state of the world between two sets (e.g., light switch up and light switch down), classifiers that detect when each behavior has been successful, and an initial hint as to where one of the behaviors will be successful. When given an image feature vector associated with a 3D location, a trained SVM predicts if the associated manipulation behavior will be successful at the 3D location. To evaluate our approach, we performed experiments with a PR2 robot from Willow Garage in a simulated home using behaviors that flip a light switch, push a rocker-type light switch, and operate a drawer. By using active learning, the robot efficiently learned SVMs that enabled it to consistently succeed at these tasks. After training, the robot also continued to learn in order to adapt in the event of failure.

Keywords Robot learning · Mobile manipulation · Home robots · Behavior-based systems · Active learning

H. Nguyen (✉) · C. C. Kemp
Healthcare Robotics Lab, Georgia Institute of Technology, Atlanta, GA, USA
e-mail: haidai@gmail.com

C. C. Kemp
e-mail: charlie.kemp@bme.gatech.edu

1 Introduction

Informing robot manipulation with computer vision continues to be a challenging problem in human environments such as homes. With homes, two types of challenges are particularly notable. First, the robot must handle wide variation in the appearance of task-relevant components of the world that can affect its ability to perform tasks successfully. Lighting can vary from home to home and from hour to hour due to indoor lighting and windows. In addition, important components of household mechanisms used during manipulation, such as drawer handles and switches, can be distinctive or even unique. The perspective from which a mobile robot observes the component will also vary.

Second, the relationship between the appearance of task-relevant components and the success or failure of a manipulation behavior is complex. For example, the mechanics of a specific device may require that the robot act at a distinct location, such as a particular drawer that needs to be pushed in the center to be closed, or a convoluted handle that the robot's gripper can only grasp at particular locations. The robot itself may also change over time and thus alter the relationship between visual appearance and a manipulation behavior, as parts of its body settle, deform, and wear.

One potential solution to these two problems is for robots to learn how specific objects respond to manipulation attempts using a behavior, and to continue to learn as they perform tasks. By using data generated through their actions without human intervention, robots can autonomously learn direct mappings from visual features to the input parameters for behaviors, enabling robust execution despite errors in calibration, pose variation, sensor noise, unexpected environmental interactions, and other factors. By continuing to learn over time, robots can also adapt to changes in the environment, the objects, and their bodies.

Published online: 01 September 2013

In this work, we present a system that enables mobile manipulators to autonomously gather data about the execution of behaviors to improve their likelihood of success in future attempts. Our work advances robot learning in three ways. First, our research addresses challenges associated with learning in scenarios that involve both mobility and manipulation. While our system does not intentionally vary the robot's base position prior to manipulation, standard navigation methods can result in significant pose variation that jeopardizes task success. We show that this issue can be resolved by directly using the robot's mobility during learning to account for this and other sources of pose variation. Notably, our methods result in predictions that implicitly account for complexities that can arise from this variation, such as alterations in the mechanism's appearance due to the robot's viewing angle, the arm having difficulty reaching a location due to the base position, or a grasp not functioning realiably due to the gripper's angle of approach.

Second, we show that autonomously learning to visually predict where a behavior will be successful can be tractable, requiring no more than a few hours to learn in real-world scenarios. By using active learning, the robots in our tests learned each visual function after fewer than 150 interactions with each device, even though the robot started from scratch and only used data it collected. The learned visual functions enabled the robots to successfully operate the devices and also have intuitive interpretations.

Third, our methods autonomously learn visual mappings for devices that have an approximately binary state, such as a light switch being up or down or a drawer being open or closed. This presents a challenge, since the robot's actions change the state of the world, which deters the robot from trying the same action again. For example, it would be difficult to learn to open a drawer if, once it is open, the robot is unable to close it. Our system addresses this difficulty by simultaneously training pairs of behaviors and alternating between them as necessary. We also formalize the ideal relationship between these pairs of behaviors and name them complementary behaviors.

We evaluated our system using an implementation on a Willow Garage PR2 robot at the Aware Home, which is a free-standing house at the Georgia Institute of Technology constructed to test new technologies. First, the robot learned to operate six devices. After learning, we tested the robot's performance in trials with each of the six devices for a total of 110 trials. In all trials, the robot autonomously operated the device successfully after at most two attempts. If the first attempt failed, the robot detected the failure and then retrained using this new negative example prior to trying a second time. We tested opening and closing drawers, turning on and off light switches, and turning on and off rocker switches. Figure 1 shows example output from the resulting trained classifiers, which classify image feature vectors as being associated with success or failure of a behavior.

Fig. 1 *Left* Willow garage PR2 operating a drawer, light switch and rocker switch using learned detector that detects regions where manipulation will succeed. *Right* Results from learned detectors during execution

2 Related work

In this section, we discuss related work in robot learning, robot manipulation, and active learning. Our work also builds on our earlier workshop publication (Nguyen and Kemp 2011).

2.1 Robot learning

Different robot learning methods such as imitation learning, interactive learning and developmental learning (Lungarella et al. 2003; Pfeifer and Scheier 1997) can be grouped by how they approach the issue of gathering data. We focus on work in which the robot learns with little human input.

2.1.1 Autonomously learning robot skills and behaviors

Learning from demonstration typically relies on substantial human involvement to acquire training data from sources such as teleoperation, shadowing, placing sensors on the demonstrator, and external observations (Argall et al. 2008). Robot reinforcement learning often involves substantial practice by a real robot, during which the real robot acquires training data. However, implementations in this area often

focus on acquiring and refining a skill, rather than applying a known skill in a new context, and typically use specially engineered environments and human intervention during training (Ijspeert et al. 2003; Kober et al. 2010; Pastor et al. 2011).

Many developmental learning systems (Lungarella and Metta 2003) use data from the robot's autonomous interactions with the environment. Much of the work to date has investigated sensorimotor coordination skills such as gaze control (Berthouze et al. 1997; Berthouze and Kuniyoshi 1998; Butko and Movellan 2010), reaching (Metta et al. 1999; Butko and Movellan 2011), pointing (Marjanovic et al. 1996), and poking (Metta and Fitzpatrick 2003). Our work focuses on complex multi-step behaviors that have direct applications in domestic settings.

2.1.2 Learning to visually detect grasps

Grasping, being a fundamental skill for mobile manipulators, have received significant attention in robot learning. Similar to our work, many have investigated associating visual features with 3D locations on objects to localize grasp frames. One of the earliest investigations in grasp learning is by Dunn and Segen (1988) that matched objects using visual features and learned grasps through trial and error. Instead of learning one grasping classifier, Zhang and Rossler's (2004) system learned separate classifiers for grasp position and grasp orientation. Saxena et al.'s (2008) method learned a classifier using a data set of simulated grasps and was successful in grasping objects in uncluttered environments. The authors in Montesano and Lopes (2009) views the same problem as one of learning object affordances and proposed a method for estimating grasp densities in images of objects on uncluttered backgrounds. Researchers in Erkan et al. (2010) learned a mapping from 3D edge features using active and semi-supervised learning. For most systems discussed, the authors gathered data manually with the exception of Saxena et al. (2008) where a simulator was used. In contrast, we present a system that can function with a variety of different behaviors, without the need for custom simulators, and in settings where hand-labeled data are not available.

2.1.3 Autonomously learning to perceive

In contrast to motor learning, most work in learning for perception relies on data captured manually (Ponce et al. 2006), captured in simulation (Klingbeil et al. 2008; Saxena et al. 2008), or downloaded from the web (http://www.semantic-robot-vision-challenge.org/, Chatzilari et al. 2011). Although large data sets can be collected from these sources, data generated can be biased (Torralba and Efros 2011) and may not match what the robot will encounter. Likewise, the relationship between these data and the robot's actions may not be clear. For example, a good location for a person to grasp or a location that a person believes would be good for grasping may not be appropriate for a particular robot. Accurate simulation of physical objects can also be hard to obtain (Abbeel et al. 2006).

The system that we present uses data generated from self-experience, (similar to Salganicoff et al. (1996), Erkan et al. (2010), Pastor et al. (2011), and Kober et al. (2010)). This has the advantage of training data that is well-matched to the particular robot and its task. However, obtaining labeled examples can be difficult, since the robot needs to act in the real-world and human labeling can be labor intensive and have errors, ambiguity, and inconsistencies (Barriuso and Torralba 2012). We address this issue in our work by combining active learning, which reduces the number of examples needed, with autonomous learning methods that eliminate the need for human labeling beyond an initialization process.

Past work in learning for perceptual categorization, a process where agents learn through interaction with the world to divide sensory information into distinct groupings, has used data from the robot's experience. However, most systems were designed to classify simple geometric objects such as cylinders and rectangles using cross-modal information (Krichmar and Edelman 2002; Coelho et al. 2001; Christian Scheier 1996).

A relatively small subset of work investigates more complex objects found in human environments. For example, Stober and Kuipers (2011) demonstrated an approach for extracting spatial and geometric information from raw sensorimotor data. Kraft et al. (2010) presented a system that gradually learns object representations and associates them with object-specific grasps. Katz and Brock (2008) showed a method with which a robot determines the structure of articulated objects through experimentation. And, van Hoof et al. (2012) presented a system that selects maximally informative actions to segment tabletop scenes.

Paolini et al.'s (2012) system uses a generative approach to estimate task success by estimating the poses of grasped objects with sensor readings then combining it with a model of task success given the estimated pose. In contrast, our approach uses discriminative modeling mapping straight from sensor readings to a label correlated with expected success. While generative modeling often resulted in a more interpretable model, discriminative modeling allows our system to be more agnostic of the particulars of behaviors used.

Previous work by the authors of Sukhoy and Stoytchev (2010) is notable for its similarity to our approach. They presented a system that uses an uncertainty sampling scheme to actively learn the appearance of doorbell buttons. In contrast, our approach uses a different active learning algorithm, works with a mobile manipulator operating in situ devices, and handles persistent change to the state of the world.

2.2 Task-relevant feature detection

In a parallel thread to robot learning, there has been recognition in the mobile manipulation community of the importance of exploiting task structure to reduce the complexity of operating in the real-world (Katz et al. 2008). Work in articulated object perception (Katz and Brock 2008), tool tip detection (Kemp and Edsinger 2006), door handle detection (Klingbeil et al. 2008), behavior-based grasping (Jain and Kemp 2010), use of a sink (Okada et al. 2006), and corner detection for towel folding (Maitin-shepard et al. 2010) suggests that low-dimensional task-specific perception can be highly-effective and that recovery of complex representations of the state of objects prior to manipulation is often unnecessary. These particular examples have used hand-coded and hand-trained feature detectors. With our approach, robots autonomously learn, after a human-aided initialization period, to classify visual features as being relevant to the success of a specific behavior or not.

2.3 Active learning and curiosity driven learning

With many robot learning scenarios, unlabeled data can be readily acquired but labeling the data is costly, a issue that can be addressed by approaches such as active (Settles 2012) and semi-supervised learning. In our work, use active learning to pick data to be labeled based a data point's value in improving the learner's model. With many active learning algorithms, at each iterative learning step the learner is given an option to select a data point to be labeled out of a set of unlabeled data points. For one class of proposed approaches, the learner picks the data point whose label it is most uncertain about (Lewis and Catlett 1994; Culotta and McCallum 2005; Settles and Craven 2008). With disagreement-based methods, learner ensembles select the data point they most disagree on Cohn et al. (1994). More computationally demanding methods, however, attempt to explicitly minimize future expected error or variance (Roy and McCallum 2001; Berger et al. 1996; Lafferty et al. 2001). There are also proposals to combine semi-supervised and active learning to exploit structure in unlabeled data (McCallum and Nigam 1998; Muslea et al. 2002; Zhu et al. 2003). Although there have been several large scale studies of active learning methods on different data sets showing its superiority over randomly picking data points for labeling (Korner and Wrobel 2006; Schein and Ungar 2007; Settles and Craven 2008), the best active learning algorithm to use in each circumstance has been application specific. In our work, we use a heuristic that picks the data point closest to the decision boundary of a support vector machine (SVM) for labeling, a method that has been shown to perform well in a variety of applications (Jain et al. 2010; Schohn and Cohn 2000; Tong and Koller 2000).

3 Approach

Our approach enables a mobile manipulator to autonomously learn a function that takes a 2D RGB image and a registered 3D point cloud as input and returns a 3D location at which a manipulation behavior is likely to succeed. To do so, it requires a pair of manipulation behaviors, verification functions that detect when each behavior has been successful, and an initial hint as to where one of the behaviors will be successful.

Each behavior must have input parameters that correspond with a 3D location that specifies where the behavior will act. During training, our system executes each behavior multiple times using different 3D locations around the device being manipulated and records whether or not the behavior succeeded at each location. For each 3D location, the system creates an image feature vector using an area of the registered 2D RGB image associated with the 3D location. These image feature vectors are labeled with whether or not the behavior succeeded or failed at their associated 3D locations. In other words, the collected data set consists of positive and negative examples of image feature vectors that were or were not associated with the success of the behavior. With a classifier trained from this data set, the robot can then predict if the associated behavior will succeed at a 3D location based on the image feature vector associated with the location.

To avoid user intervention during training, our procedure trains two behaviors at the same time, switching to the other behavior when the current behavior succeeds. This enables our method to operate devices that can be approximated as having two binary states, such as a drawer being open or closed. Using a pair of behaviors allows the robot to change the device back and forth between these two states, so that training can continue autonomously. For example, instead of training a drawer opening behavior in isolation, our process flips to training a drawer closing behavior when opening succeeds and vice versa until the classifier converges. We also formalize the relationship between the two behaviors and define them as complementary behaviors.

Using self-generated data takes considerable time, since each labeled image feature vector requires that the robot execute the behavior at a 3D location and observe the results. To avoid needing an intractable number of trials, our method uses active learning to execute the behavior at an informative 3D location at each iteration. Specifically, our procedure trains a support vector machine (SVM) after each trial using the current labeled data. It then uses a heuristic proposed by Schohn and Cohn (2000) to select the unlabeled image feature vector that is closest to the current SVM's decision boundary to be labeled next. It then executes the behavior at the 3D location associated with this image feature vector.

Our training procedure has two phases. The first is an initialization phase where the user selects the behavior pair

to train, gives a seed 3D location, and positions the robot's mobile base for training. The next phase is an autonomous phase where the SVM active learning procedure runs until the learner converges. After convergence, each behavior has a classifier that predicts 3D locations where it will succeed.

During runtime, if the behavior's verification function detects a failed attempt, our procedure appends this negative example to the data set, retrains the classifier, and tries again using the output of this new classifier (Sect. 3.3.3).

In the following sections, we discuss the requirements of our learning procedure (Sect. 3.1), properties of complementary behaviors (Sect. 3.2), our training procedure in detail (Sect. 3.3), and classification infrastructure (Sect. 3.4).

3.1 Requirements

Our methods make the following assumptions:

1. The robot can execute a set of behaviors, $\{B_1, \ldots B_n\}$, where each behavior, B_i, requires a 3D location, p_{3D}, in the robot's frame of reference as initial input. We have previously demonstrated that behaviors of this form can perform a variety of useful mobile manipulation tasks when provided with a 3D location designated with a laser pointer (Nguyen et al. 2008).
2. The robot has a way of reliably detecting whether or not a behavior it has executed was successful or not. Specifically, a verification function, V, returns whether or not a behavior succeeded. For this work, V takes the form $V(I(b), I(a))$, where $I(x)$ is the array of robot sensor readings when the state of the world is x. The states b and a are the states before and after the robot executes a behavior.
3. For each behavior, B, there is a complementary behavior, B^*. If B successfully executes, then successful execution of B^* will return the world to a state that allows B to execute again. We discuss the implications of this requirement in the next section (Sect. 3.2).
4. Upon successful execution, a behavior returns a 3D location near where its complementary behavior can successfully execute.

3.2 Complementary behaviors

In order to train without human intervention our procedure uses a complementary pair of behaviors during its data gathering process. We introduce the notion of a complementary robot behavior B^* to a behavior B as being a behavior that is capable of "reversing" the state of the world, so that behavior B can be used again. For example, if behavior B's function is to turn off the lights using a light switch, its complement, B^*, would turn the lights back on using that light switch. If a behavior opens a door, then its complement would close the door.

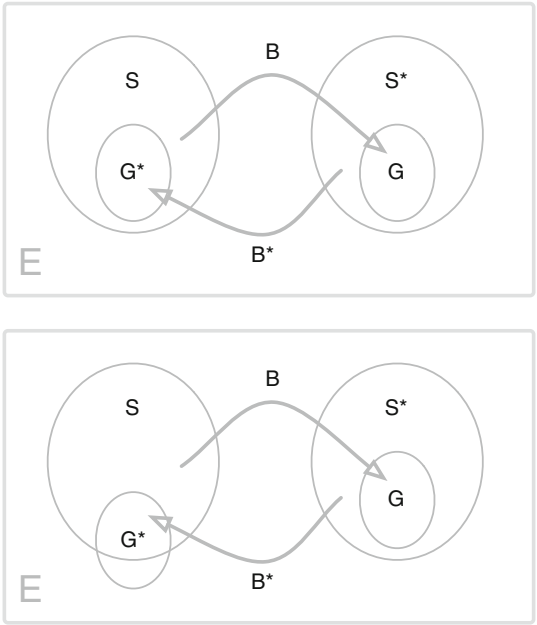

Fig. 2 Relationships between set S, G, S^*, G^*, B, and B^*. *Top* an example set of complementary behaviors where $G^* \subseteq S$ and $G \subseteq S^*$. In this case, the effect of B is reversible using B^*. *Bottom* an example set of behaviors that are not complements with $G^* \nsubseteq S$, so B^* can produce states that are not in S

We formalize our notion of complementary behaviors by defining the relationship between ideal complementary behaviors. We first define a hypothetical state space E that contains the states of everything in the world, including the robot's state. We then represent execution of behavior B given an initial state of the world $i \in E$ as $B(i)$, where B is an operator that takes the initial state of the world i as input and returns the resulting state of the world $r \in E$. Furthermore, when B is applied to a state $s \in S$, where $S \subseteq E$ is a set of **s**tarting states, it returns $g \in G$, where $G \subseteq E$ is a set of goal states. We define

$$G = \{g | V(I(i), I(g)) = \text{success} \wedge g = B(i) \wedge i \in E\} \quad (1)$$

and

$$S = \{s | g \in G \wedge g = B(s) \wedge s \in E\}. \quad (2)$$

Intuitively, if the state of the world, s, is a start state, $s \in S$, then the behavior B will be successful and the resulting state of the world, $g = B(s)$, will be a goal state, $g \in G$.

We now define a complement B^* of behavior B to have a set of start states, S^*, and a set of goal states, G^*, such that $G^* \subseteq S$ and $G \subseteq S^*$ (see Fig. 2). This guarantees that applying B's complement, B^*, after successfully applying B will result in a state of the world that allows B to once

Fig. 3 Illustration of the initialization procedure for a pair of behaviors that flip light switches. *Left* position robot in front of the switch. *Middle* illuminate an initial 3D location as input to the behavior using a laser pointer. *Right* A 3D location associated with success (*green*) and a 3D location associated with failure (*red*) after initialization. Candidate 3D locations to explore are shown in white (Color figure online)

again be applied successfully. More formally, it guarantees that $B^*(B(i)) \in S$ when $i \in S$, and that $B(B^*(i)) \in S^*$ when $i \in S^*$.

3.3 Autonomous training

3.3.1 Initialization

Our initialization procedure is motivated by the scenario in which a user would take the robot on a home tour and point out 3D locations using a green laser pointer (Nguyen et al. 2008) and specify behaviors applicable to those locations. After this tour, the robot would later autonomously navigate back and learn to robustly perform the behaviors.

For this paper, we have implemented an initialization procedure that starts with the user navigating the robot to be in front of the device to be operated using a gamepad interface. Then using a green laser pointer (Nguyen et al. 2008), the user designates an initial 3D location to begin exploring by holding the laser point steady at the intended target. The robot samples 3D points around this designated location (using a spherical Gaussian with a variance of 4 cm) and executes the behavior pair with respect to them. After each execution of a behavior at a 3D location, the behavior's verification function returns a label of either success or failure. The sampling process continues until the procedure gathers data points from at least one successful and one failed trial. These two data points are then used to train SVM classifiers that guide the data gathering process with the active learning heuristic (Schohn and Cohn 2000).

After this initialization, the robot stores a 2D mobile base pose with respect to a global map, the user provided 3D location, an SVM trained using two labeled data points, and labels indicating which pair of behaviors is applicable at the specified location. We illustrate this procedure in Fig. 3. In addition, the user navigates the robot to eight different poses in the room, referred to as practice poses, each at least a half meter away from the device. The robot also stores the 2D mobile base poses associated with these eight practice poses.

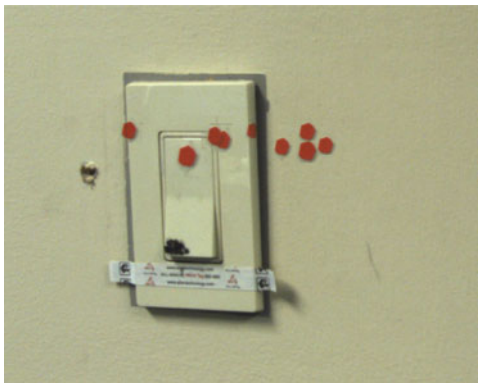

Fig. 4 This figure shows a visualization of task variation due to the robot's mobility. We affixed a *red dot* at the center of a rocker switch. The robot attempted to navigate to the same pose and take the same picture of the switch ten times. This image superimposes the *red dot* from nine images onto the first image to illustrate the wide variation due to navigation. One of the ten dots is obscured by two others. The switch plate shown has a width of 7.0 cm. If the robot were to use its localization estimate to press this switch, most of the attempts would result in a failure (Color figure online)

3.3.2 Training procedure

Our training procedure is designed to emulate conditions that the robot would encounter when performing the task. After receiving a command, the robot navigates to the device so that it can execute the commanded behavior. Navigation and localization errors result in variations that can substantially reduce the performance of a behavior, such as variation in the robot's point of view. We illustrate task variation due to navigation in Fig. 4. Our training method samples from this source of task variation by commanding the robot to navigate to one of eight practice poses in the room and then commanding it to navigate back to the device (see Fig. 5).

After navigating to the device, our procedure begins an active learning phase (see Fig. 6). We summarize this phase in Algorithm 1. The process starts with the robot capturing an RGB image and a registered 3D point cloud. The robot then computes image feature vectors for 3D points randomly sampled from the point cloud around the device

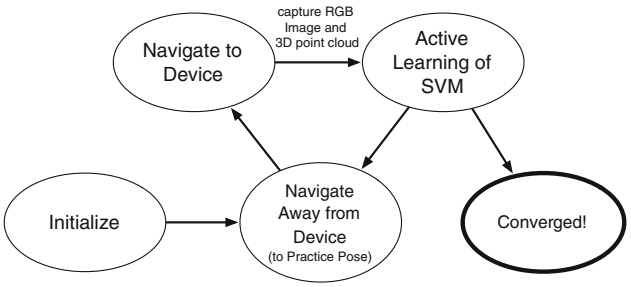

Fig. 5 Overview of the training procedure: initialization of the classifier; specification of practice poses in the environment by the user; and a loop that navigates the robot to each practice pose and back to the device until the robot gathers enough training data

Algorithm 1: practice($point^{3D}$, behavior, comp_behavior, stop_criteria)

```
instances, candidates^3D = extract_features(point^3D);
while True do
    instance, candidate^3D = svm_pick(behavior, instances,
    candidates^3D);
    if stop_criteria(behavior) or svm_converged(behavior,
    instances) then
        break;
    end
    success, candidate^3D* = execute_behavior(behavior,
    candidate^3D);
    add_instance_and_retrain_svm(instance, success);
    instances = instances \ instance;
    candidates^3D = candidates^3D \ candidate^3D;
    if success then
        practice(candidate^3D*, comp_behavior, None,
        stop_criteria=stop_on_first_success);
    end
end
```

(*extract_features*). It then iteratively selects image feature vectors (*svm_pick*) that it labels by executing the behavior at the associated 3D location and using the verification function (*execute_behavior*). After each execution, the process retrains the SVM classifier with a data set that incorporates the newly acquired labeled example (*add_instance_and_retrain_svm*). In order to make our implementation efficient, the robot does not collect a new RGB image or registered 3D point cloud between these executions. If an execution succeeds (*If(success)*), however, the robot begins this same process with the complementary behavior, which first captures a new RGB image and registered 3D point cloud. When executing the complementary behavior, the system does so until it succeeds with no limits on the number of retries.

The procedure stops after gathering a maximum of six labeled image feature vectors or the learner converges (*stop_criteria*). We imposed this conservative maximum limit, determined heuristically, because image feature vectors gathered from the same view are correlated, which can confuse the learning heuristic and result in the training process stopping prematurely. However, if we pick a number that is too small, the robot's base would move more often lengthening the training time.

This process continues until *svm_converge* is satisfied for each of the eight practice poses. Once it is satisfied for a particular practice pose, the robot no longer navigates to the pose. We define convergence for a practice pose to occur when after driving up to the device from the practice pose, none of the initially computed image feature vectors are closer to the decision boundary than the current support vectors.

3.3.3 Behavior execution procedure

The training process above produces a classifier that can reliably detect locations where the associated behavior will succeed. To use this classifier, our robot navigates to the device using the 2D map pose stored during initialization, classifies 3D points in the view that it sees, finds the mode of

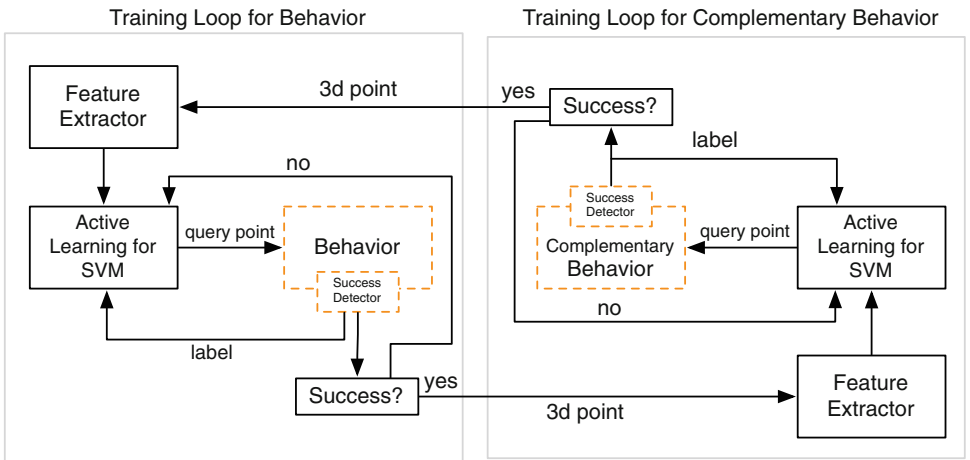

Fig. 6 Illustration of the classifier training procedure where the system trains the complementary behavior upon success of the first behavior and vice versa. *Dashed orange boxes* on the two behaviors and success detectors highlight that these modules are provided as input to our system (Color figure online)

the positive classified points using kernel density estimation, selects the 3D point in the point cloud closest to this mode, and executes the associated behavior using the resulting 3D location.

If the behavior fails to execute using this 3D location, our procedure adds the associated image feature vector as a negative example to the data set and retrains the classifier. This new example changes the classifier's decision boundary. The robot then selects a new 3D location using the retrained classifier with the originally computed image feature vectors. This continues until the behavior is successful. It then adds the image feature vector associated with this success to the data set as a positive example and retrains the SVM. In contrast to systems where the execution process is independent of data gathering and training, the robot has the opportunity to retrain its classifier when it detects errors made during execution, giving the possibility of lifelong training.

3.4 Classification

Our methods use standard support vector machine (SVM) classifiers trained with fully-labeled data. Our current implementation uses batch learning and retrains these classifiers many times. Our datasets tended to be small with fewer than 200 examples, which made this feasible, but online learning with appropriate classifiers might achieve comparable performance with improved efficiency.

We denote the data for our classification problem as $D = \{(x_1, y_1), \ldots (x_N, y_N)\}$ where $x_i \in \mathbb{R}^M$ is the feature vector that represents the 2D appearance of 3D point i and $y_i \in \{1, -1\}$ is the label that represents success, 1, or failure, -1, of the behavior when it was executed with 3D point i as input. Our goal is to produce a classifier that predicts y_j for future instances of x_j encountered by the robot.

As functional structures on many household devices are often small compared to nonfunctional components, such as the size of a switch relative to the plate or wall, there is typically an unbalanced data set problem, since there can be many more negative than positive examples. In unbalanced data sets the SVM can return trivial solutions that misclassify all the positive samples, since the misclassification cost term in the SVM objective is defined over all samples. To prevent this issue, we use an SVM formulation that separates the costs of misclassifying the negative class from the cost of misclassifying the positive class (Chang and Lin 2001),

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C^+ \sum_{y_i=1} \xi_i + C^- \sum_{y_i=-1} \xi_i \\ s.t. \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, i = 1, \ldots, l, \end{aligned} \tag{3}$$

where $\mathbf{w}$ and b are SVM parameters, ξ_i counts the margin violations for misclassified points (in the case of nonseparable data), and $\phi()$ is the radial basis kernel function we use (discussed in Sect. 4.1).

This formulation separates the SVM misclassification cost scalar C into C^+ and C^- which are, respectively, costs due to negative and positive misclassifications. For our system, we set C^- to be 1, and C^+ to be the number of negative examples over the number of positive examples. This scaling keeps the percentage of misclassified positive and negative examples similar in our skewed data set, where there might be many more negative than positive examples. Without this adjustment, we found that training often returned trivial classifiers that classified any input vector as negative.

3.4.1 Active learning heuristic

Our training process iteratively builds a data set that it uses to train the classifier. Before each trial, the system selects the image feature vector to label. To select the feature vector, the system uses a heuristic developed in Schohn and Cohn (2000) that selects the feature vector closest to the decision boundary of the existing SVM, under the condition that it is closer to the boundary than the SVM's support vectors. The procedure converges when no feature vectors remain that are closer to the decision boundary than the support vectors.

At each iteration i of our procedure, we define the previous iteration's data set as D_{i-1}, the current set of support vectors as $X_i^{sv} = \{x_1^{sv}, \ldots, x_P^{sv}\}$, the unlabeled image feature vectors as $X_i^q = \{x_1^q, \ldots, x_M^q\}$, and the SVM distance function, which measures distance to the decision boundary, as $d(\mathbf{x_i}) = \left|\mathbf{w}^T\phi(\mathbf{x}_i) + b\right|$. The system selects the unlabeled image feature vector that is closest to the decision boundary as specified by the following expression:

$$\underset{\mathbf{x}_i^q : \forall \mathbf{x}_j^{sv}\ d(\mathbf{x}_i^q) < d(\mathbf{x}_j^{sv})}{\operatorname{argmin}} \quad d(\mathbf{x}_i^q) \tag{4}$$

3.4.2 Features

The feature generation procedure, which is illustrated in Fig. 7, takes as input a 3D point cloud, a registered high resolution RGB image, and a reference 3D point. The system first selects random 3D points from the point cloud, without replacement, around the reference 3D point according to a Gaussian distribution $\mathcal{N}(\bar{p}, \Sigma)$, where $\Sigma = \text{diag}(v_x, v_y, v_z)$ with v_x, v_y, and v_z being, respectively, variances in the x, y, and z direction. The Gaussian mean $\bar{\mathbf{p}}$ is set to the 3D reference point. This Gaussian search prior enables the system to save computational effort and focus its attention on the device that the robot is supposed to manipulate.

After randomly selecting a set of 3D points, the system projects each 3D point $\mathbf{p}_i^c$ into the high resolution RGB image, $proj(\mathbf{p}_i^c)$. For each projected 3D point, it collects square image patches of successively increasing size cen-

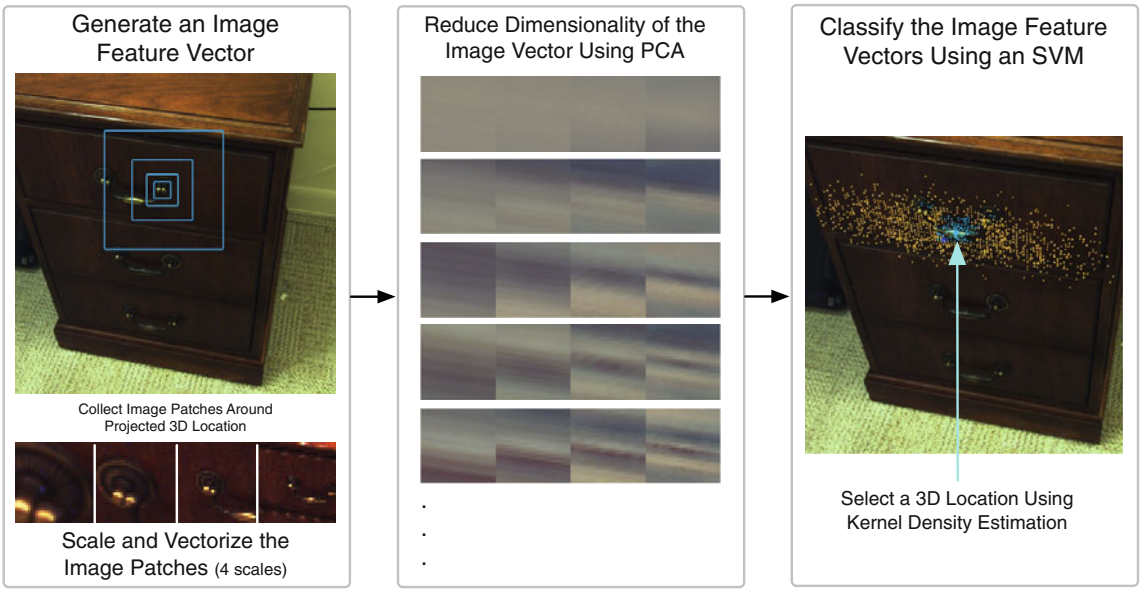

Fig. 7 To select a 3D location at which the behavior is likely to be successful, the system first generates image feature vectors for a set of 3D locations. It does so by vectorizing and then reducing the dimensionality of scaled image patches centered around the 2D projection of each 3D location. Then it uses an autonomously trained SVM to classify each of these image feature vectors as predicting success (*blue*) or failure (*orange*) of the behavior. Finally, it selects a specific 3D location using kernel density estimation (Color figure online)

tered at the projected 2D point in the RGB image, scales these patches to have the same height and width, vectorizes them, and concatenates them into an image feature vector. The system then uses Principle Components Analysis (PCA) to reduce the dimensionality of these image feature vectors. We discuss the specifics of these steps in Sect. 4.1.

4 Implementation

4.1 Learner parameters

We implemented our system on a PR2 robot: a mobile manipulator produced by Willow Garage with two arms, an omnidirectional base, and a large suite of sensors. Our system uses 3D point clouds and 5 megapixel RGB images from the robot's tilting laser range finder and Prosilica camera.

Starting with a 3D point cloud and registered RGB image, our process randomly selects 3D points from the point cloud as described in Sect. 3.4.2. For each selected 3D point, the system collects image patches at 4 scales centered around the point's 2D projection in the RGB image. The raw image patches have widths of 41, 81, 161, and 321 pixels. They are then scaled down to 31×31 pixel image patches, vectorized, and concatenated into an 11,532 element image feature vector for each 3D point. Computer vision researchers have used similar representations. We selected these particular sizes by hand while developing the system. Computational limitations, the resolution of the robot's camera images, and the appearance of the mechanisms influenced our selection, but we would not expect system performance to be sensitive to these sizes. The vectors are then reduced to 50 element vectors by projecting them onto PCA basis vectors that are calculated for each action using the 11,532 element image feature vectors computed from the first 3D point cloud and RGB image captured during initialization. We used 50 PCA vectors, which preserved 99 % of the variance across 20 images of a light switch and drawer.

To classify these 50 dimensional image feature vectors, we use SVMs with radial basis function kernels. We set the hyperparameters of this kernel using an artificially labeled data set. We created this data set by taking ten different 3D point clouds and RGB images of a light switch from different views and geometrically registered them. After hand-labeling one 3D point cloud and RGB image, we geometrically propagated labels to the other nine. To find the kernel hyperparameters, we split the labeled image feature vectors from this data set into a training set and a test set. Finally, we performed a grid search (Chang and Lin 2001) for the set of hyperparameters that best generalized to unseen data in the test set.

4.2 Behaviors

To evaluate our system, we implemented three pairs of complementary behaviors that operate light switches, rocker switches and drawers. These tasks are sensitive to the location at which an action is performed. For example, light switches

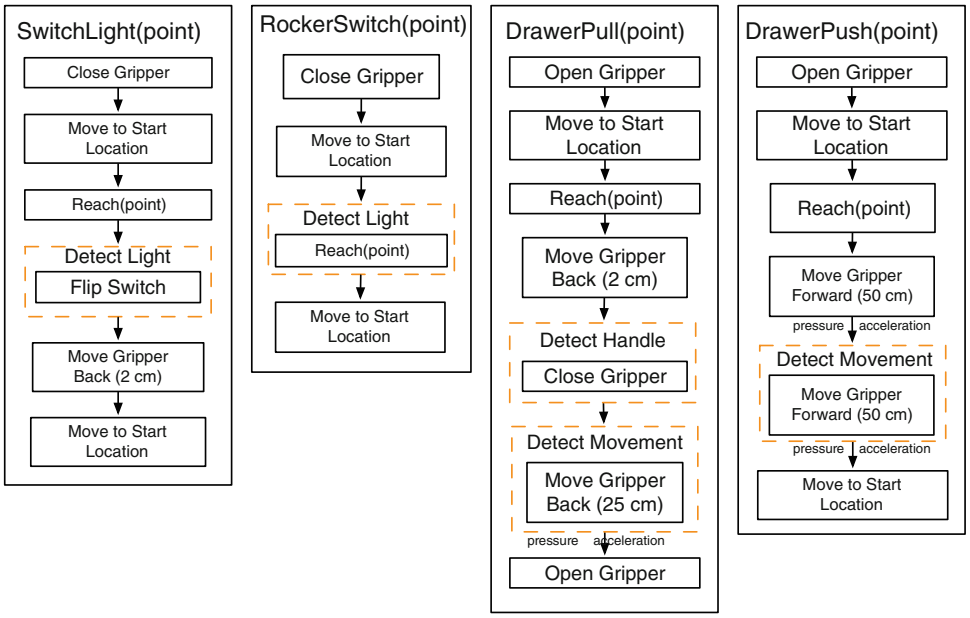

Fig. 8 Sequence of actions performed by each of the eight behaviors used in this work for operating a light switch, rocker switch and drawer. *Dotted orange boxes* indicate procedures for detecting success or failure in a given behavior (Color figure online)

are small targets that require high precision and accuracy for the PR2 to operate with its finger tips. As illustrated in Fig. 4, we have found that a PR2 will rarely succeed at flipping a light switch if it simply navigates to a pre-recorded location and moves the arm through a pre-recorded motion without visual feedback.

4.3 Light switch behaviors

Our light switch behavior's strategy is to reach forward to the specified 3D location, stop on contact detected with gripper tip tactile sensors, then slide along the contacted surface in the direction of the switch. A successful 3D location needs to place the robot's finger so that its width will make contact with the switch and far enough above or below the switch so that the finger will move the switch down or up. Figure 8 shows the sequence of actions taken by this behavior.

The behavior starts with the robot closing its gripper (Close Gripper), moving the gripper to a pre-manipulation location (Move to Start Location), reaching to the given 3D location (Reach), flipping the switch by sliding along the flat surface (Flip Switch), moving the gripper back (Move Gripper Back), then moving back to the initial location (Move to Start Location).

There are a few steps in this behavior where the robot detects tactile events. When reaching, the robot stops when it detects contact using pressure sensors on its finger tips. Next, the sliding movement stops after detecting a spike in acceleration with the accelerometer embedded in the robot's gripper. In the context of this task, this spike in acceleration typically corresponds with the light switch flipping.

To detect success, our behavior measures the difference between the average intensity of an image captured before sliding along the surface and an image captured after. A large difference indicates that the lighting intensity changed.

The complementary behavior is identical except for a change in the direction of flipping. After executing, the behavior and complementary behavior return the 3D location input with a predefined offset (± 8 cm).

4.4 Rocker switch behaviors

Our rocker switch behavior consists solely of a reaching out step similar to the light switch behavior above, since the force applied from contact during the reach procedure is enough to activate the switch. A successful 3D location will result in the robot's fingers pushing in the top or bottom of the rocker switch.

This behavior uses the same image differencing method to detect success as the light switch behavior. It calculates the difference between images captured before and after the robot reaches forward. After executing, the behavior and complementary behavior return the 3D location with a predefined offset (± 5 cm).

4.5 Drawer behaviors

Pulling open and pushing closed a drawer require different behaviors and success detection methods. Our pulling behavior reaches to the drawer handle location, detects contact, moves back slightly, grasps with the reactive grasper from Hsiao et al. (2010), and pulls. When pulling, failure is

detected if the grasp fails or the robot fails to pull for at least 10 cm while in contact with the handle. A successful 3D location will result in the robot's gripper grasping the handle well enough to pull it back by at least 10 cm. When pushing, failure is detected if the gripper does not remain in contact with the surface for at least 10 cm. This classifies events where the robot pushes against a closed drawer or an immovable part of the environment as failures. After executing, the behavior and complementary behavior return the 3D location the tip of the gripper was in immediately after pulling or pushing.

4.6 Robot controllers used

We now discuss the various controllers used for sub-behaviors shown in Fig. 8. For the arms, we had the option to use either a joint or Cartesian controller. The joint controller takes as input a set of 7 joint angles for each arm and creates splines for each joint individually to move the arm smoothly to the given goal. Built on Nakanishi et al.'s (2005) acceleration Jacobian pseudo-inverse control scheme, the Cartesian trajectory controller (provided by the ROS package robot_mechanism_controllers) attempts to keep the robot's end-effector as close as possible to a given 6 degree-of-freedom (DoF) goal. As the arms possess 7 DoF, this controller allows the remaining one degree-of-freedom to be specified using a posture goal consisting of a set of 7 joint angles. Finally, the Cartesian controller allows for much more compliant motions since it is less concerned about following exact joint level goals.

At the beginning and end of most behaviors discussed, we issue a "Move to Start Location" command that internally calls the Cartesian controller to move to a preset 3D point then uses the joint controller to fix the arm stiffly in place. For the close and open gripper command, we use the pr2_gripper_action package in ROS to move to a fully opened or fully closed position while limiting the maximum effort. To implement "Reach", we send a series of Cartesian goals to the arm in a straight line starting from the end-effector's current position and ending at the given goal with the option to stop when detecting readings from either the pressure of acceleration sensors. Finally, for modules that move the gripper backward or forward for a given distance, we again use the Cartesian controller but instead of linearly interpolating to the goal, we just send the final goal point. These motions also have the option of stopping based on tactile events detected by the robot's grippers.

5 Evaluation

We evaluated our system using six separate devices. We first tested on a rocker switch using the PR2 robot named GATSBII in our lab, the Healthcare Robotics Lab (HRL). For the remaining five devices we performed tests in the Georgia Tech Aware Home, a residential lab on campus used as a test bed for new technologies.

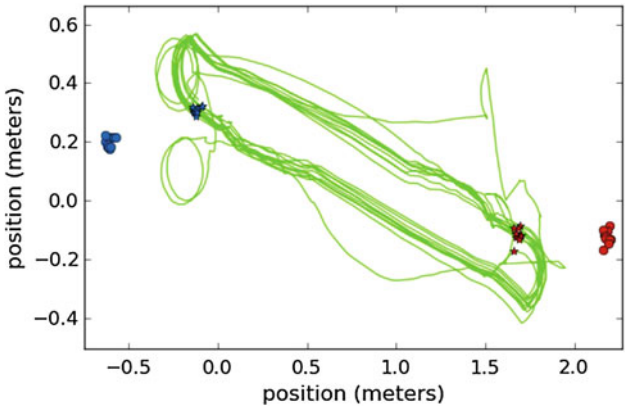

Fig. 9 Results of experiments for which we used a motion capture system to track the robot's pose while navigating between two goal poses (*blue* and *red*). *Green* is the path the robot took. *Stars* indicate the final poses of the robot after it navigated to the goal poses. *Circles* show a point 50 cm in front of the robot (Color figure online)

In each environment, we began our evaluation by creating an occupancy grid map of the area with the PR2's built-in navigation package (Marder-Eppstein et al. 2010). Then, after initialization (Sect. 3.3.1), we ran the autonomous training system (Sect. 3.3.2) until convergence. The experimenter provided eight practice poses, four for each behavior in the pair. Here, we picked four for each behavior as it allowed us to exhaustively pick locations from which the robot would travel to the mechanism in most rooms. We have not observed the training process to be particularly sensitive to this parameter however. The training system ran without experimenter intervention except for pausing and resuming when the robot's batteries ran low. In all, we trained 12 classifiers, a result of having six devices and a pair of behaviors for each device ($12 = 6 \times 2$).

After finishing the training sessions, we evaluated each classifier by running each behavior multiple times, giving 110 trials in all ($110\,\text{trials} = (5\,\text{devices} \times 2\,\text{behaviors} \times 10\,\text{trials}) + (1\,\text{device} \times 2\,\text{behaviors} \times 5\,\text{trials})$). During each trial we allowed the behavior to retry and incorporate information from failures if it did not succeed the first time. However, we discarded any data gathered during the retry procedure by previous trials at the start of each new trial to obtain accurate error statistics for the original classifier.

For the devices we used in our evaluation, the functional components are difficult for the PR2's laser range finder to detect. Light switches only show up as a few protruding 3D points similar to other noisy 3D points produced by the sensor. The rocker switch appears as a flat 2D texture on the 3D point cloud. Drawer handles tend to be metallic and reflective resulting in an absence of 3D points. Using features from RGB images enabled the robot to overcome these challenges.

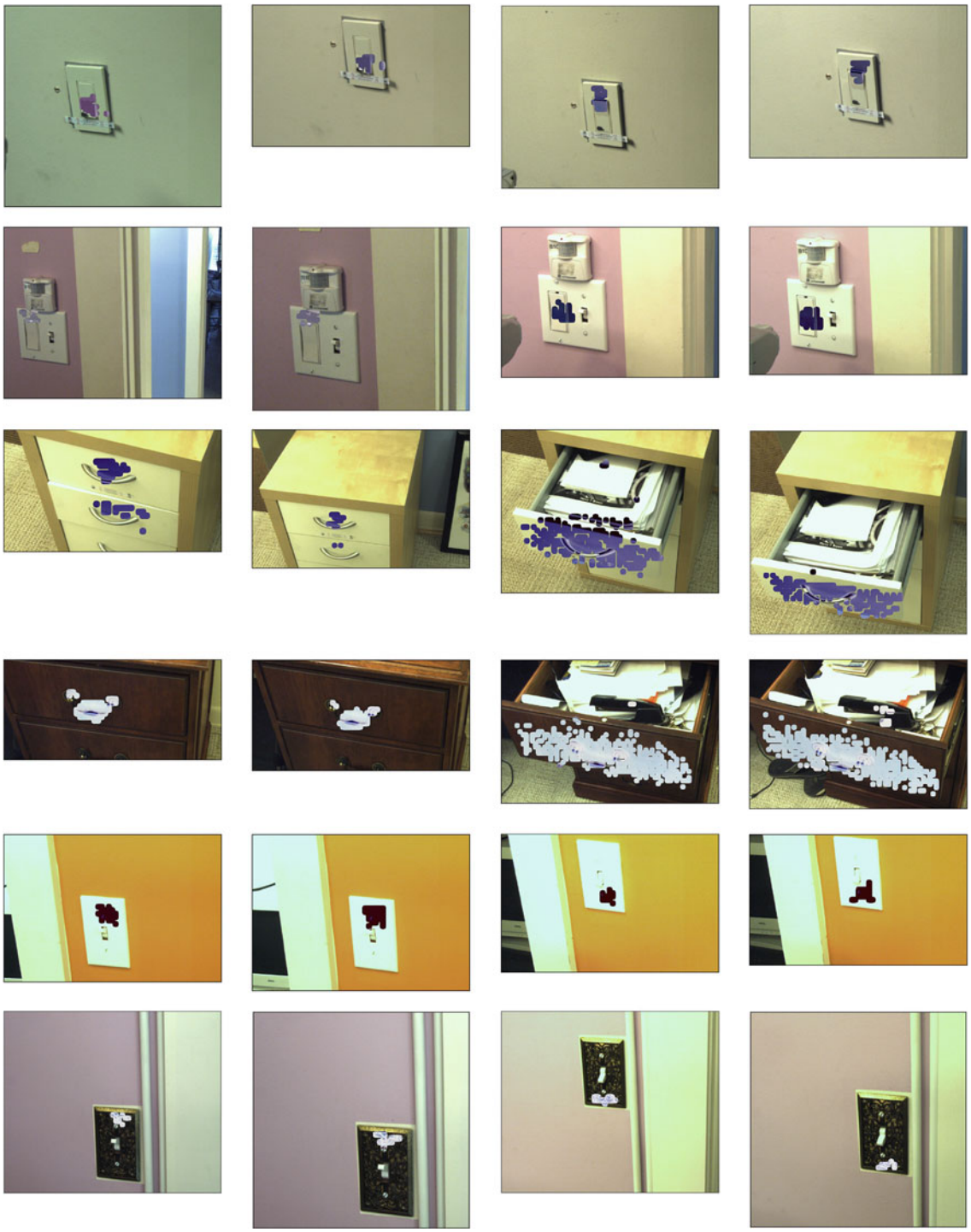

Fig. 10 Each pair of images shows classification results of learned detectors just after convergence then on a new test image. Areas with *inverted colors* mark locations identified as leading to success of associated behaviors. To show these results, we took positively predicted points, placed those points in a grid, then overlaid this grid on the original image. *Row 1* detectors for a rocker switch in our lab. *Row 2* detectors for a different rocker switch in the Aware Home. *Row 3* detectors for pushing and pull a wooden drawer. *Row 4* detectors for another dark wooden drawer. *Row 5* detectors for a regular light switch. *Row 6* detectors for an ornate light switch

5.1 Effects of navigation errors

To better understand the variation in the task due to the robot's mobility, we investigated how the pose of the PR2 varies when navigating to a goal pose. Using a room equipped with a NaturalPoint OptiTrak motion capture system, we tracked the pose of the PR2 and commanded the robot to navigate back and forth to two goal poses ten times each. As the standard

deviation of the robot's Cartesian position does not represent angular errors, we calculated errors for a point 50 cm in front of the robot, which is representative of where a device would be located. The standard deviation of the location in front of the robot was 1.85, and 1.79 cm in the x and y directions, respectively. For the second position, the standard deviation was 1.55 and 2.38 cm in the x and y directions, respectively. We show the results of this experiment in Fig. 9. These errors demonstrate that navigating to a pre-recorded location and moving the arm through a pre-recorded motion would result in large variation that can result in failure. For example, the robot's finger tips are 2.0 cm wide and light switches are only 0.8 cm wide.

6 Results

Figure 10 shows the locations that the trained SVMs predict will be likely to lead to the success of their associated behaviors. These predictions are solely a function of the visual appearance of each location as represented by its image feature vector. These visualizations of the classifier output demonstrate that the classifiers identify locations relevant to their associated behaviors. For example, the robot autonomously discovers that opening a drawer requires grasping at the location of the drawer handle, while closing a drawer can be performed across the front surface of the drawer. The visualizations also show that different drawer handles can have distinct task-relevant properties. For example, the opening behavior works best when grasping the middle of the silver handle, but can succeed by grasping the far ends of the brass handle.

Due to the distribution for random sampling including some points on the lower handles for the white drawers, the SVM estimates that success can be achieved by pulling on the top handle or the bottom handle. The illustrates a limitation with our current approach, since the verification function for pulling a drawer open can not tell the difference between the top or the bottom drawer. It also shows the influence of the distribution used to randomly sample 3D locations. At the same time, it suggests that the visual classifiers may have some ability to generalize to distinct objects.

For the light switches, the behaviors slide along the surface of the switch. The robot autonomously discovered that locations that are along the switch plate above and below the switch are likely to lead to success. Additionally, it does not predict success for locations along the wall, which is appropriate since the robot's fingers get caught on the switch plate edge if the robot tries to slide along the wall to the switch.

In Table 1, we show the number of examples collected for each classifier. The median number of examples needed was 77, and the maximum needed was 145 examples. With the rocker switch, where examples are noisy due to the middle of the switch being an unreliable spot to push, the number of examples increased to 145 indicating a sensitivity of our approach to label noise.

Table 1 Training examples (abbreviated Ex.) gathered for each action

Action	Positive ex.	Negative ex.	Total
HRL rocker on	49	96	145
HRL rocker off	47	94	141
Aware H. rocker on	26	47	73
Aware H. rocker off	29	52	81
Ikea drawer open	23	35	58
Ikea drawer close	23	39	62
Brown drawer open	21	62	83
Brown drawer close	25	46	71
Orange switch on	17	43	60
Orange switch off	20	31	51
Ornate switch on	38	66	104
Ornate switch off	40	76	116

Table 2 For each trained behavior we ran ten trials. We list the number of tries until success for these trials below

Action	1st Try	2nd Try
HSI rocker on	2	3
HSI rocker off	4	1
Aware home rocker on	10	
Aware home rocker off	9	1
Ikea drawer open	10	
Ikea drawer close	10	
Brown drawer open	10	
Brown drawer close	10	
Orange switch on	8	2
Orange switch off	9	1
Ornate switch on	9	1
Ornate switch off	9	1

Table 2 shows the results of using these trained classifiers after training. Encouragingly, over the 110 trials our behavior execution process attained a 100 % success rate after at most two tries. In addition, errors that led to retries usually caused the robot to miss an appropriate location on the device by a small distance.

7 Future work

There are a number of potential extensions to this work, and interesting issues left to consider. Although we have picked a particular active learning framework, other frameworks might perform better. Our current system depends on the verification function properly labeling the success or failure of an attempt, both for training data and switching to the com-

plementary behavior. Reducing this dependence or finding ways to learn or adapt the verification function automatically could be worthwhile. In addition, we assume that each device is completely new to the robot, but many devices of a particular class have visual similarities. Data from other devices might provide a prior and reduce the training required. Similarly, the structure of successful locations might be shared across devices, even if they are visually distinct. For example, the front surfaces of drawers often being pushable, the centers of drawers often being pullable, and the centers of light switch panels often being switchable could be useful information, even if aspects of their appearances change dramatically.

8 Discussion and conclusions

In general, there are risks for a robot that learns in human environments and an unrestrained learning system can get into situations that are dangerous to itself, to the environment, or to people. We address this issue by limiting the robot to using a few classes of behaviors in parts of the home that users have designated as safe for robot learning. Additionally, the behaviors move the robot's arm compliantly and use haptic sensing to decide when to stop moving. By learning in situ, a robot's data gathering activities do not have to stop after its training phase and can potentially continue for as long as the robot remains in service.

Autonomous learning in human environments is a promising area of research that gives robots methods to cope with devices that they have not encountered before and many forms of real-world variation. We have presented methods that enable a mobile manipulator to autonomously learn to visually predict where manipulation attempts might succeed. As we discussed in the introduction, our work advances autonomous robot learning in three ways. First, our approach uses a robot's mobility as an integral part of autonomous learning, which enables the robot to handle the significant task variation introduced by its mobility. Second, our research demonstrates that by using active learning, a robot can autonomously learn visual classifiers solely from self-generated data in real-world scenarios with a tractable number of examples. Third, our research introduces complementary behaviors to address challenges associated with autonomously learning tasks that change the state of the world.

Acknowledgments We thank Aaron Bobick, Jim Rehg, and Tucker Hermans for their input. We thank Willow Garage for the use of a PR2 robot, financial support, and other assistance. This work was funded in part by NSF awards CBET-0932592, CNS-0958545, and IIS-1150157.

References

Abbeel, P., Quigley, M., & Ng, A. Y. (2006). Using inaccurate models in reinforcement learning. In *International Conference on Machine Learning* (pp. 1–8).

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, *57*, 469–483.

Barriuso, A., & Torralba, A. (2012). Notes on image annotation. In *MIT Technical Note*.

Berger, A. L., Pietra, V. J. D., & Pietra, S. A. D. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, *22*, 39–71.

Berthouze, L., Bakker, P., & Kuniyoshi, Y. (1997). Learning of oculomotor control: A prelude to robotic imitation. In *International Conference on Robotics and Intelligent Systems*.

Berthouze, L., & Kuniyoshi, Y. (1998). Emergence and categorization of coordinated visual behavior through embodied interaction. *Machine Learning*, *5*, 369–379.

Butko, N. J., & Movellan, J. R. (2010). Developmental robotics architecture for active vision and reaching. In *International Conference on Development and Learning* (pp. 1–6).

Butko, N. J., & Movellan, J. R. (2010). Learning to look. In *International Conference on Development and Learning* (pp. 70–75).

Chang, C.-C., & Lin, C.-J. (2001). *Libsvm: A library for support vector machines*. http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Chatzilari, E., Nikolopoulos, S., Papadopoulos, S., & Kompatsiaris, C. Z. Y. (2011). Semi-supervised object recognition using flickr images. In *International Workshop on Content-Based Multimedia Indexing* (pp. 229–234).

Christian Scheier, D. L. (1996). Categorization in a real-world agent using haptic exploration and active perception. In *International Conference on Simulation of Adaptive Behavior* (pp. 65–75). Cambridge, MA: MIT Press.

Coelho, J., Piater, J., & Grupen, R. (2001). Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. *Robotics and Autonomous Systems*, *37*, 195–218.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, *15*, 201–221.

Culotta, A., & McCallum, A. (2005). Reducing labeling effort for structured prediction tasks. In *Conference on Artificial Intelligence (AAAI)* (Vol. 2, pp. 746–751).

Dunn, G., & Segen, J. (1988). Automatic discovery of robotic grasp configuration. In *International Conference on Robotics and Automation* (Vol. 1, pp. 396–401).

Erkan, A., Kroemer, O., Detry, R., Altun, Y., Piater, J., & Peters, J. (2010). Learning probabilistic discriminative models of grasp affordances under limited supervision. In *IEEE International Conference on IROS* (pp. 1586–1591).

Hsiao, K., Chitta, S., Ciocarlie, M. & Jones, G. (2010). Contact-reactive grasping of objects with partial shape information. In *IROS*.

Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2003). Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 1523–1530).

Jain, A., & Kemp, C. C. (2010). El-e: An assistive mobile manipulator that autonomously fetches objects from flat surfaces. *Autonomous Robots*, *28*, 45–64.

Jain, P., Vijayanarasimhan, S. & Grauman, K. (2010). Hashing hyperplane queries to near points with applications to large-scale active learning. In *Advances in Neural Information Processing Systems (NIPS)*

Katz, D., & Brock, O. (2008). Manipulating articulated objects with interactive perception. In *ICRA* (pp. 272–277).

Katz, D., Kenney, J., & Brock, O. (2008). How can robots succeed in unstructured environments? In *RSS, Robot Manipulation Workshop*.

Kemp, C. C. & Edsinger, A. (2006). Robot manipulation of human tools: Autonomous detection and control of task relevant features. In *ICDL*.

Klingbeil, E., Saxena, A., & Ng, A. Y. (2008). Learning to open new doors. In *RSS Workshop on Robot Manipulation*.

Kober, J., Oztop, E., & Peters, J. (2010). Reinforcement learning to adjust robot movements to new situations. In *RSS*.

Korner, C. & Wrobel, S. (2006). Multi-class ensemble-based active learning. In *European Conference on Machine Learning (ECML)* (pp. 687–694).

Kraft, D., Detry, R., Pugeault, N., Baeski, E., Guerin, F., Piater, J., & Krger, N. (2010). Development of object and grasping knowledge by robot exploration. In *IEEE Transactions on Autonomous Mental Development* (pp. 368–383).

Krichmar, J. L., & Edelman, G. M. (2002). Machine psychology: Autonomous behavior, perceptual categorization and conditioning in a brain-based device. *Cerebral Cortex*, *12*, 818–830.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)* (pp. 282–289).

Lewis, D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *International Conference on Machine Learning (ICML)* (pp. 148–156).

Lungarella, M., & Metta, G. (2003). Beyond gazing, pointing, and reaching: A survey of developmental robotics. In *EPIROB* (pp. 81–89).

Lungarella, M., Mettay, G., Pfeiferz, R., & Sandini, G. (2003). Developmental robotics: A survey. *Connection Science*, *15*, 151–190.

Maitin-Shepard, J., Cusumano-Towner, M., Lei, J. & Abbeel, P. (2010). Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *ICRA*.

Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B. P., & Konolige, K. (2010). The office marathon: Robust navigation in an indoor office environment. In *International Conference on Robotics and Automation*.

Marjanovic, M., Scassellati, B., & Williamson, M. (1996). Self-taught visually-guided pointing for a humanoid robot. In *International Conference on Simulation of Adaptive Behavior* (pp. 35–44).

McCallum, A., & Nigam, K. (1998). Employing em in pool-based active learning for text classification. In *International Conference on Machine Learning (ICML)* (pp. 350–358).

Metta, G., & Fitzpatrick, P. (2003). Early integration of vision and manipulation. *Adaptive Behavior*, *11*, 109–128.

Metta, G., Sandini, G., & Konczak, J. (1999). A developmental approach to visually-guided reaching in artical systems. In *Neural Networks* (Vol. 12, pp. 1413–1427).

Montesano, L. & Lopes, M. (2009). Learning grasping affordances from local visual descriptors. *ICDL* (pp. 1–6).

Muslea, I., Minton, S., & Knoblock, C. (2002). Active + semi-supervised learning = robust multi-view learning. In *International Conference on Machine Learning (ICML)* (pp. 435–442).

Nakanishi, J., Mistry, M., & Schaal, S. (2005). Comparative experiments on task space control with redundancy resolution. In *IEEE International Conference on Intelligent Robots and Systems* (pp. 1575–1582).

Nguyen, H., Jain, A., Anderson, C., & Kemp, C. C. (2008). A clickable world: Behavior selection through pointing and context for mobile manipulation. In *IROS*.

Nguyen, H., & Kemp, C. C. (2011). Autonomous active learning of task-relevant features for mobile manipulation. In *RSS 2011 Workshop on Mobile Manipulation: Learning to Manipulate*.

Okada, K., Kojima, M., Sagawa, Y., Ichino, T., Sato, K., & Inaba, M. (2006). Vision based behavior verification system of humanoid robot for daily environment tasks. In *Humanoid Robots* (pp. 7–12).

Paolini, R., Rodriguez, A., Srinivasa, S. S. , & Mason, M. T. (2012). A data-driven statistical framework for post-grasp manipulation. In *International Symposium on Experimental Robotics*.

Pastor, P., Kalakrishnan, M., Chitta, S., Theodorou, E. & Schaal, S. (2011). Skill learning and task outcome prediction for manipulation. In *ICRA*.

Pfeifer, R., & Scheier, C. (1997). Sensory-motor coordination: The metaphor and beyond. *Robotics and Autonomous Systems*, *20*, 157–178.

Ponce, J., Berg, T., Everingham, M., Forsyth, D., Hebert, M., Lazebnik, S., et al. (2006). Dataset issues in object recognition. In *Toward Category-level Object Recognition* (Vol. 4170, pp. 29–48).

Roy, N., & McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. In *International Conference on Machine Learning (ICML)* (pp. 441–448).

Salganicoff, M., Ungar, L. H., & Bajcsy, R. (1996). Active learning for vision-based robot grasping. *Machine Learning*, *23*, 251–278.

Saxena, A., Driemeyer, J., & Ng, A. Y. (2008). Robotic grasping of novel objects using vision. *IJRR*, *27*, 157–173.

Schein, A., & Ungar, L. (2007). Active learning for logistic regression: An evaluation. *Machine Learning*, *68*, 235–265.

Schohn, G. & Cohn, D. (2000). Less is more: Active learning with support vector machines. In *ICML* (pp. 839–846).

Scipy gaussian kde function. http://www.scipy.org/doc/api_docs/SciPy.stats.kde.gaussian_kde.html. Accessed 17 Jan 2012.

Semantic robot vision challenge. (2007). http://www.semantic-robot-vision-challenge.org/. Accessed 17 Jan 2012.

Settles, B. (2012). *Active learning*. San Rafael, CA: Morgan & Claypool Publishers.

Settles, B., & Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1070–1079).

Stober, J., & Kuipers, R. M. B. (2011). Learning geometry from sensorimotor experience. In *International Conference on Development and, Learning*.

Sukhoy, V., & Stoytchev, A. (2010). Learning to detect the functional components of doorbell buttons using active exploration and multimodal correlation. In *IEEE International Conference on Humanoid Robots*.

Tong, S., & Koller, D. (2000). Support vector machine active learning with applications to text classification. In *Conference on Machine Learning (ICML)* (Vol. 2, pp. 45–66).

Torralba, A., & Efros, A. (2011). Unbiased look at dataset bias. In *IEEE Conference on computer Vision and Pattern Recognition (CVPR)* (pp. 1521–1528).

van Hoof, H., Kroemer, O., Amor, H. B., & Peters, J. (2012). Maximally informative interaction learning for scene exploration. In *IROS* (pp. 5152–5158).

Willow garage. http://www.willowgarage.com. Accessed 17 Jan 2012.

Zhang, J., & Rossler, B. (2004). Self-valuing learning and generalization with application in visually guided grasping of complex objects. *Robotics and Autonomous systems*, *47*, 117–127.

Zhu, X., Lafferty, J., & Ghahramani, Z. (2003). Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML Workshop on the Continuum from Labeled to Unlabeled Data*.

Hai Nguyen Ph.D. Candidate at the Georgia Institute of Technology. He is a member of the Center for Robotics and Intelligent Machines (RIMG@GT), and the Healthcare Robotics Lab. He received a Bachelor of Science in Computer Science from Georgia Tech in 2006. His research interests include home robotics, autonomous mobile manipulation, and machine learning for robots.

Charles C. Kemp is an associate professor at the Georgia Institute of Technology in the Department of Biomedical Engineering, and has adjunct appointments in the School of Interactive Computing and the School of Electrical and Computer Engineering. He earned a doctorate in Electrical Engineering and Computer Science (2005), an M.Eng., and B.S. from MIT. He founded the Healthcare Robotics Lab in 2007.